Research Article

An Enhanced Artificial Bee Colony Algorithm with Solution Acceptance Rule and Probabilistic Multisearch

Alkin Yurtkuran and Erdal Emel

Department of Industrial Engineering, Uludag University, Görükle Campus, 16059 Bursa, Turkey

Correspondence should be addressed to Alkin Yurtkuran; alkin@uludag.edu.tr

Received 3 July 2015; Accepted 9 September 2015

Academic Editor: Ezequiel López-Rubio

Copyright © 2016 A. Yurtkuran and E. Emel. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The artificial bee colony (ABC) algorithm is a popular swarm based technique, which is inspired from the intelligent foraging behavior of honeybee swarms. This paper proposes a new variant of ABC algorithm, namely, enhanced ABC with solution acceptance rule and probabilistic multisearch (ABC-SA) to address global optimization problems. A new solution acceptance rule is proposed where, instead of greedy selection between old solution and new candidate solution, worse candidate solutions have a probability to be accepted. Additionally, the acceptance probability of worse candidates is nonlinearly decreased throughout the search process adaptively. Moreover, in order to improve the performance of the ABC and balance the intensification and diversification, a probabilistic multisearch strategy is presented. Three different search equations with distinctive characters are employed using predetermined search probabilities. By implementing a new solution acceptance rule and a probabilistic multisearch approach, the intensification and diversification performance of the ABC algorithm is improved. The proposed algorithm has been tested on well-known benchmark functions of varying dimensions by comparing against novel ABC variants, as well as several recent state-of-the-art algorithms. Computational results show that the proposed ABC-SA outperforms other ABC variants and is superior to state-of-the-art algorithms proposed in the literature.

1. Introduction

Optimization techniques play an important role in the field of science and engineering. Over the last five decades, numerous algorithms have been developed to solve complex optimization algorithms. Since more and more present-day problems turn out to be nonlinear, multimodal, discontinuous, or dynamic in nature, derivative-free, nonexact solution methods attract ever-increasing attention. Evolutionary biology or swarm behaviors inspired most of these methods. There have been several classes of algorithms proposed in this evolutionary or swarm intelligence framework including genetic algorithms [1, 2], memetic algorithms [3], differential evolution (DE) [4], ant colony optimization (ACO) [5], particle swarm optimization (PSO) [6], artificial bee colony algorithm (ABC) [7], cuckoo search [8], and firefly algorithm [9].

The ABC is a biologically inspired population-based metaheuristic algorithm that mimics the foraging behavior of honeybee swarms [7]. Due to its simplicity and ease of application, the ABC has been widely used to solve both continuous and discrete optimization problems since its introduction [10]. It has been shown that ABC tends to suffer poor intensification performance on complex problems [11–13]. To improve the intensification performance of ABC, many researchers have focused on the search rules as they control the tradeoff between diversification and intensification. Diversification means the ability of an algorithm to search for unvisited points in the search region, whereas intensification is the process of refining those points within the neighborhood of previously visited locations to improve solution quality. Various new search strategies, mostly inspired from PSO and DE, have been proposed in the literature. Zhu and Kwong [14] proposed a global best guided ABC, which utilizes the global best individual's information within the search equation similar to PSO. Gao et al. [15] introduced another variant of global best ABC. Inspired by DE, Gao and Liu [13] introduced a modified version of
the ABC in which ABC/Best/1 and ABC/Rand/1 were employed as local search equations. Kang et al. [16] described the Rosenbrock ABC, which combines Rosenbrock's rotational method with the original ABC. To improve diversification, Alatas [11] employed chaotic maps for initialization and chaotic searches within a search strategy. Akay and Karaboga [17] introduced a modified version of the ABC in which frequency of perturbation operator is controlled adaptively and the ratio of variance operator was introduced. Liao et al. [18] proposed a detailed experimental analysis and comparison of an ABC variant with different search equations. Gao et al. [19] introduced two new search equations for onlooker and employed bee phases and a new robust comparison technique for candidate solutions. ABC. Qiu et al. [20] were inspired from DE/current-to-best/1 strategy in DE algorithm and proposed a modified ABC. Banitalebi et al. [21] proposed an enhanced compact ABC, which did not store the actual population of candidate solution; instead their approach employed probabilistic representation. Wang et al. [22] presented multistrategy ABC, in which a pool of different search strategies was constructed and various search strategies were used during the search process. Gao et al. [23] introduced a bare bones ABC with parameter adaptation and fitness-based neighborhood to improve the intensification performance of standard ABC. Ma et al. [24] reduced the redundant search moves and maintained the diversity of the swarm by introducing hybrid ABC with life cycle and social learning. Furthermore, ABC has been successfully applied to solve various types of optimization problems, such as production scheduling [25, 26], vehicle routing [27], location-allocation problem [28], image segmentation [29], wireless sensor network routing [30], leaf-constrained minimum spanning tree problem [31], clustering problem [32], fuel management optimization [33], and many others [34–36]. Readers can refer to Karaboga et al. [10] for an extensive literature review of the ABC and its applications.

This study presents an enhanced ABC with solution acceptance rule and probabilistic multisearch (ABC-SA) in order to solve global optimization problems efficiently. In ABC-SA, three search mechanisms with different diversification and intensification characteristics are employed. Moreover, search mechanism selection probabilities \( p_{s1}, p_{s2}, \) and \( p_{s3} \) are introduced to control the balance between diversification and intensification. In our proposed approach, a search mechanism is established using the selection probabilities to generate a new neighbor solution from the current one. Additionally, a solution acceptance rule is implemented, in which not only better solutions but also worse solutions may be accepted by using a probability function. A nonlinearly decreasing acceptance probability function is employed, thus allowing worse solutions to be more likely accepted in the early phases of the search. Therefore, ABC-SA algorithm explores the search space more widespread, especially in the early phases of the search process. By using solution acceptance rule and implementing different search mechanisms of contrasting nature, ABC-SA balances the trade-off between diversification and intensification efficiently. The proposed approach is tested on six different benchmark functions with varying dimensions and compared to novel ABC, PSO, and DE variants. Computational results reveal that ABC-SA outperforms competitor algorithms in terms of solution quality.

The main contributions of the proposed study are as follows:

(i) Three different search mechanisms are employed with varying diversification and intensification abilities. Probabilistic multisearch with predetermined probability values are employed to determine the search mechanism to be used to generate candidate solutions. Therefore, ABC-SA explores and exploits the search space efficiently.

(ii) Instead of a greedy selection, a new candidate solution acceptance rule is integrated, where a worse solution may have a chance to be accepted as new solution. By the help of this new acceptance rule, ABC-SA achieves better diversification performance, specifically in the early phases of the search.

The remainder of this paper is structured as follows: Section 2 presents the traditional ABC; Section 3 introduces the proposed framework; the instances, parameter settings, and computational results are presented in Section 4 and finally Section 5 concludes the paper.

2. Artificial Bee Colony Algorithm

The ABC has inspired from the organizational nature and foraging behavior of honeybee swarms. In the ABC algorithm, the bee colony comprises three kinds of bees: employed bees, onlooker bees, and scout bees. Each bee has a specialized task in the colony to maximize the nectar amount that is stored in the hive. In ABC, each food source is placed in the \( D \)-dimensional search space and represents a potential solution to the optimization problem. The amount of nectar in the food source is assumed to be the fitness value of a food source. Generally, the number of employed and onlooker bees is the same and equal to the number of food sources.

Each employed bee belongs to a food source and is responsible for mining the corresponding food source. Then, employed bees pass the nectar information to onlooker bees in the “dance area.” Onlooker bees wait in the hive and select a food source to mine based on the information coming from the employed bees. Here, more beneficial food sources will have higher selection probabilities to be selected by onlooker bees. In ABC, in order to decide if a food source is abandoned or not, trial counters and a predetermined limit parameter are used. If a solution represented by a food source does not improve during a number of trials (limit), the food source is abandoned. When the food source is abandoned, the corresponding employed bee will become a scout bee and randomly generate a new food source and replace it with the abandoned one.

The ABC algorithm consists of four main steps: initialization, employed bee phase, onlooker bee phase, and scout bee phase. After the initialization step, the other three main steps of the algorithm are carried out repeatedly in a loop.
Computational Intelligence and Neuroscience

3

until the termination condition is met. The main steps of the ABC algorithm are as follows.

Step 1 (initialization). In the initialization step, the ABC generates a randomly distributed population of SN solutions (food sources), where SN also denotes the number of employed or onlooker bees. Let \( x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\} \) represent the \( i \)th food source, where \( D \) is the problem size. Each food source is generated within the limited range of \( j \)th index by

\[
x_{i,j} = x_{j}^{\text{min}} + \varphi_{i,j} \left( x_{j}^{\text{max}} - x_{j}^{\text{min}} \right),
\]

where \( i = 1, 2, \ldots, \text{SN} \), \( j = 1, 2, \ldots, D \), \( \varphi_{i,j} \) is a uniformly distributed random real number in \([0,1]\), and \( x_{j}^{\text{min}} \) and \( x_{j}^{\text{max}} \) are the lower and upper bounds for the dimension \( j \), respectively. Moreover, a trial counter for each food source is initialized.

Step 2 (employed bee phase). In the employed bee phase, each employed bee visits a food source and generates a neighboring food source in the vicinity of the selected food source. Employed bees search a new solution, \( v_{i,j} \), by performing a local search around each food source \( i = 1, 2, \ldots, \text{SN} \) as follows:

\[
v_{i,j} = x_{i,j} + \phi (x_{i,j} - r_{1,j}),
\]

where \( j \) is a randomly selected index \( j \in \{1, 2, \ldots, D\} \) and \( r_1 \in \{1, 2, \ldots, \text{SN}\} \) is a randomly chosen food source that is not equal to \( i \); that is, \( r_1 \neq i \). \( \phi \) is a random number within the range \([-1, 1]\) generated specifically for each \( i \) and \( j \) combination. A greedy selection is applied between \( x_i \) and \( v_j \) by selecting the better one.

Step 3 (onlooker bee phase). Unlike the employed bees, onlooker bees select a food source depending on the probability value \( p_a \), which is determined by nectar amount associated with that food source. The value of \( p_a \) is calculated for \( i \)th food source as follows:

\[
p_i = \frac{\text{fit}_i}{\sum_{j=1}^{\text{SN}} \text{fit}_j},
\]

\[
\text{fit}_i = \begin{cases} 1 & f_i \geq 0 \\ 1 + \text{abs}(f_i) & f_i < 0, \end{cases}
\]

where \( \text{fit}_i \) is the fitness value of solution \( i \) and calculated as in (4) for minimization problems. Different fitness functions are employed for maximization problems. By using this type of roulette wheel based probabilistic selection, better food sources will more likely be visited by onlooker bees. Therefore, onlooker bees try to find new candidate food sources around good solutions. Once the onlooker bee chooses the food source, it generates a new solution using (2). Similar to the employed bee phase, a greedy selection is carried out between \( x_i \) and \( v_j \).

Step 4 (scout bee phase). A trial counter is associated with each food source, which depicts the number of tries that the food source cannot be improved. If a food source cannot be improved for a predetermined number of tries (limit) during the onlooker and employed bee phases, then the employed bee associated with that food source becomes a scout bee. Then, the scout bee finds a new food source using (1). By implementing the scout bee phase, the ABC algorithm easily escapes from minimums and improves its diversification performance.

It should be noted that, in the employed bee phase, a local search is applied to each food source, whereas in the onlooker bee phase better food sources will more likely be updated. Therefore, in ABC algorithm, the employed bee phase is responsible for diversification whereas the onlooker bee phase is responsible of intensification. The flow chart of the ABC is given in Figure 1.

3. Proposed Framework

In this section, the proposed algorithm is described in detail. First, a solution acceptance rule is presented. Second, a novel probabilistic multisearch mechanism is proposed. Finally, the complete ABC-SA mechanism is given.

3.1. Solution Acceptance Rule. In order to strengthen the diversification ability of ABC-SA mechanism, a solution acceptance rule is proposed. Instead of greedy selection in both employed and onlooker bee phases, an acceptance probability is given to worse solutions. The main idea behind this acceptance probability is not to restrict the search moves to only better solutions. By accepting a worse solution, the procedure may escape from a local optimum and explore the search space effectively. In ABC-SA algorithm, if a worse solution is generated, it is accepted if the following condition holds:

\[
r < p_a
\]

\[
p_a = p_0 \frac{1 + \cos((\text{iter}/\text{Max.iter}) \pi)}{2},
\]

where \( r \) is a random real number within \([0,1]\), \( p_a \) is the acceptance probability, \( p_0 \) denotes the initial probability, and \( \text{iter} \) and \( \text{Max.iter} \) represent the current iteration number and the maximum iteration number, respectively. According to (6), the acceptance probability \( p_a \) is nonlinearly decreased from \( p_0 \) to zero during the search process. As can be seen from (6), \( p_a = 0 \) when \( \text{iter} = \text{Max.iter} \) and the range of \( p_a \) is \([0, p_0]\). A typical \( p_a \) graph is given in Figure 2 and Algorithm 1 presents the implementation of the solution acceptance rule. At this point, it is important to note that the trial counter is incremented, whether a worse candidate solution is accepted or not.

3.2. Probabilistic Multisearch Strategy. In standard ABC, a candidate solution is generated using the information of the parent food source with the guidance of the term
\( \phi_{i,j}(x_i, x_j) \) in (3). However, there is no guarantee that a better individual influences the candidate solution; therefore, it is possible to have a poor convergence speed and intensification performance. In fact, studying search equations is a trending topic to improve the ABC’s performance. Recently, numerous search equations have been proposed, such as [13–16, 19, 20, 37, 38]. It is well known that the balance between diversification and intensification is the most critical part of any metaheuristic algorithm.

In ABC-SA approach, instead of employing a single search mechanism throughout the search process, a probabilistic multisearch mechanism with three different search rules is used. A probabilistic selection is employed using predefined probability parameters to select the search rule.
within both employed and onlooker bee phases. The three search rules which were proposed by [7, 13, 14], respectively, are presented as follows:

\[ v_{i,j} = x_{i,j} + \phi (x_{i,j} - x_{r1,j}), \]  
(7)

\[ v_{i,j} = x_{i,j} + \phi (x_{i,j} - x_{r1,j}) + \psi (x_{\text{best,j}} - x_{i,j}), \]  
(8)

\[ v_{i,j} = x_{\text{best,j}} + \phi (x_{i,j} - x_{r1,j}), \]  
(9)

where \( i \) is a food source, \( j \) is a randomly selected index for all \( i = 1, 2, \ldots, SN \), and \( j \in \{1, 2, \ldots, D\} \), respectively. \( r1 \) is a randomly chosen food source where \( r1 \neq i \). \( \text{best} \) stands for the global best solution, whereas \( \text{best} \) is the best solution in the current population. \( \phi \) represents a real random number within the range of \([-1,1]\) [7]. Finally, \( \psi \) is a real random number within the range of \([0,C]\) where \( C \) is a predetermined number [13].

Equation (7) is the original search rule, which was discussed in previous sections. Equation (8) is presented to improve the intensification capability of ABC. Equation (8) uses the information provided by the global best solution which is similar to PSO. In (9), \( \text{best} \) guides the search with the random effect of the term \( \phi (x_{i,j} - x_{r1,j}) \). Equation (7) has an explorative character, whereas (9) favors intensification. On the other hand, (8) explores the search space using the second term and exploits effectively by the third term. Therefore, (8) balances diversification and intensification performance. In summary, the proposed ABC-SA uses three different search rules to achieve a trade-off between diversification and intensification. In ABC-SA, search probabilities \( p_{s1}, p_{s2}, \) and \( p_{s3} \) are introduced such that \( \sum p_{sk} = 1 \) and \( k = 1, 2, 3 \) to select a search rule to be used in the employed and the onlooker bee phases. A roulette wheel method is employed with three cumulative ranges \( 0 \leq p_{s1} \leq p_{s1}, 0 \leq p_{s2} < p_{s3} - p_{s2}, \) and \( 0 \leq p_{s3} < p_{s3} - p_{s2} \) assigned to (7), (8), and (9), respectively, where \( p_{s3} = 1 \). Algorithm 2 shows the mechanism of probabilistic multisearch.

3.3. Proposed Approach. Algorithm 3 summarizes the ABC-SA framework. The novel parts of the ABC-SA mechanism are the probabilistic multisearch and the solution acceptance rule (Lines 9 and 19) and the solution acceptance rule (Lines 10 and 20) sections.

4. Computational Results

4.1. Test Instances. In literature, many test functions with different characters were used to test algorithms [11, 13, 15, 17, 19–21, 34, 37–40]. Unimodal functions have one local minimum as the global optimum. These functions are generally used to test the intensification ability of algorithms. Multimodal functions have one or more local optima which may be the global optimum. Therefore, diversification behavior of algorithms is analyzed on multimodal instances. Separable functions can be written as sum of \( n \) functions with one variable, whereas nonseparable functions cannot be reformulated as subfunctions. In this study, to analyze the performance of the proposed ABC-SA algorithm, 13 scalable benchmark functions with dimensions \( D = 50, D = 100, \) and \( D = 200 \) are used and listed in Table 1. They are Rosenbrock, Ackley, Rastrigin, Weierstrass, Schwefel 2.26, Shifted Sphere, Shifted Schwefel 1.2, Shifted Rosenbrock, Shifted Rastrigin, Step, Penalized 2, and Alpine. In Table 1, function label, name, formulation, type (UN: unimodal and nonseparable, MS: multimodal and separable, and MN: multimodal and nonseparable), range, and optimal values \( f(x^*) \) are given.

4.2. Parameters Settings. Parameter settings may have a great influence on the computational results. The ABC-SA mechanism has seven control parameters such as maximum iteration number (Max.iter), limit, population size (SN), \( p_{s1}, p_{s2}, \) and \( p_{s0} \). Maximum iteration number is the termination condition, and \( p_{s0} \) is the initial acceptance probability. First, Max.iter is set to 4,000, limit = \( 0.2 \times D \times SN \), where \( D \) is the dimension of the problem [21], SN is taken to be 40 for 50D and 100D problems and 50 for 200D problems [40], and \( \psi \) is set to be a random real number within \((0,1.5)\) [14]. Then,
preliminary experiments were conducted with appropriate combinations of the following parameter values to determine the best settings:

\[ p_0 = 0.25, 0.20, 0.15, 0.10, \text{and} 0.05, \]
\[ p_{s1} = 0.2, 0.4, \text{and} 0.6, \]
\[ p_{s2} = 0.2, 0.4, \text{and} 0.6, \]
\[ p_{s3} = 0.2, 0.4, \text{and} 0.6. \]

From the results of the pilot studies, \( p_0 = 0.10, \) \( p_{s1} = 0.20, \) \( p_{s2} = 0.60, \) and \( p_{s3} = 0.20 \) settings achieved the best results. Therefore, these parameter settings are used for further experiments.

4.3. Comparison with ABC Variants. In this section, aforementioned ABC-SA is implemented and evaluated by benchmarking with other well-known ABC variants including...
<table>
<thead>
<tr>
<th>Label</th>
<th>Name</th>
<th>Formulation</th>
<th>Type</th>
<th>Range</th>
<th>$f(x^*)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Rosenbrock</td>
<td>$f_1(\bar{X}) = \sum_{i=1}^{D-1} \left[ 100(x_{i+1}^2 - x_i^2) + (x_i - 1)^2 \right]$</td>
<td>UN</td>
<td>$[-2.048, 2.048]^D$</td>
<td>0</td>
</tr>
<tr>
<td>F2</td>
<td>Ackley</td>
<td>$f_2(\bar{X}) = -20 \exp(-0.2\sqrt{\sum_{i=1}^{D} x_i^2}) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e$</td>
<td>MS</td>
<td>$[-32.768, 32.768]^D$</td>
<td>0</td>
</tr>
<tr>
<td>F3</td>
<td>Rastrigin</td>
<td>$f_3(\bar{X}) = \sum_{i=1}^{D} [x_i^2 - 10 \cos(2\pi x_i) + 10]$</td>
<td>MS</td>
<td>$[-5.12, 5.12]^D$</td>
<td>0</td>
</tr>
<tr>
<td>F4</td>
<td>Griewank</td>
<td>$f_4(\bar{X}) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$</td>
<td>MN</td>
<td>$[-600, 600]^D$</td>
<td>0</td>
</tr>
<tr>
<td>F5</td>
<td>Weierstrass</td>
<td>$f_5(\bar{X}) = \sum_{i=1}^{D} \left[ \sum_{k=0}^{k_{\max}} \sum_{b=0}^{2} a^k \cos(2\pi b (x_i + 0.5)) \right] - \sum_{k=0}^{k_{\max}} \sum_{b=0}^{2} a^k \cos(2\pi b 0.5)$, $a = 0.5$, $b = 3$, $k_{\max} = 20$</td>
<td>MS</td>
<td>$[-0.5, 0.5]^D$</td>
<td>0</td>
</tr>
<tr>
<td>F6</td>
<td>Schwefel 2.26</td>
<td>$f_6(\bar{X}) = 418.9829 \times D - \sum_{i=1}^{D} -x_i \sin\left(\sqrt{</td>
<td>x_i</td>
<td>}\right)$</td>
<td>MS</td>
</tr>
<tr>
<td>F7</td>
<td>Shifted Sphere</td>
<td>$f_7(\bar{X}) = \sum_{i=1}^{D} z_i^2 - f_{\text{bias}}$, $z = x - o$, $f_{\text{bias}} = -450$</td>
<td>US</td>
<td>$[-100, 100]^D$</td>
<td>$f_{\text{bias}}$</td>
</tr>
<tr>
<td>F8</td>
<td>Shifted Schwefel 1.2</td>
<td>$f_8(\bar{X}) = \sum_{i=1}^{D} \left( \sum_{j=1}^{D} z_j \right) + f_{\text{bias}}$, $z = x - o$, $f_{\text{bias}} = -450$</td>
<td>UN</td>
<td>$[-100, 100]^D$</td>
<td>$f_{\text{bias}}$</td>
</tr>
<tr>
<td>F9</td>
<td>Shifted Rosenbrock</td>
<td>$f_9(\bar{X}) = \sum_{i=1}^{D-1} \left( 100 (z_{i+1}^2 - z_i^2) + (z_i - 1)^2 \right) + f_{\text{bias}}$, $z = x - o + 1$, $f_{\text{bias}} = 390$</td>
<td>MN</td>
<td>$[-100, 100]^D$</td>
<td>$f_{\text{bias}}$</td>
</tr>
<tr>
<td>F10</td>
<td>Shifted Rastrigin</td>
<td>$f_{10}(\bar{X}) = \sum_{i=1}^{D} [x_i^2 - 10 \cos(2\pi x_i) + 10] + f_{\text{bias}}$, $z = x - o$, $f_{\text{bias}} = -330$</td>
<td>MS</td>
<td>$[-5, 5]^D$</td>
<td>$f_{\text{bias}}$</td>
</tr>
<tr>
<td>F11</td>
<td>Step</td>
<td>$f_{11}(\bar{X}) = \sum_{i=1}^{D} (</td>
<td>x_i + 0.5</td>
<td>^2)$</td>
<td>US</td>
</tr>
<tr>
<td>F12</td>
<td>Penalized 2</td>
<td>$f_{12}(\bar{X}) = \frac{1}{10} \left[ \sin^2(\pi x_i) + \sum_{i=1}^{D-1} (x_{i+1} - 1)^2 \left[ 1 + \sin^2(3\pi x_{i+1}) \right] + (x_i - 1)^2 \right]$</td>
<td>MN</td>
<td>$[-50, 50]^D$</td>
<td>0</td>
</tr>
<tr>
<td>F13</td>
<td>Alpine</td>
<td>$f_{13}(\bar{X}) = \sum_{i=1}^{D} [x_i \cdot \sin(x_i) + 0.1 \cdot x_i]$</td>
<td>MS</td>
<td>$[-10, 10]^D$</td>
<td>0</td>
</tr>
</tbody>
</table>
The parameters of test algorithms are set to their original values given in their corresponding papers, except for the maximum number of function evaluations, population size, and limit, which are set to the same values for all ABC variants. ABC-SA, ABC, and GABC implement random initialization mechanisms whereas IABC employs a chaotic initialization as described in [13]. All algorithms have been simulated in MATLAB environment and executed on the same computer with Intel Xeon CPU (2.67 GHz) and 16 GB of memory.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.

The computational results are presented in Table 2 for 50D problems, Table 3 for 100D problems, and Table 4 for 200D problems. In Tables 2–4, results are given in terms of mean and standard deviation of the objective values due to the repetitive runs for the global best solutions. All algorithms were run 30 times with random seeds and the stopping criteria set to 4,000 iteration, which means that 320,000 function evaluations for 50D and 100D problems and 400,000 function evaluations for 200D problems approximately. For a precise and pairwise comparison, statistical significances of the differences between the means of two algorithms are analyzed using $t$-tests where significance level is set to 0.05. In Tables 2–4, “+” in the columns next to competing algorithms shows that the competitor algorithm is better than ABC-SA and the compared algorithm is not statistically significant, and “−” indicates that the difference between the ABC-SA and the compared algorithm is statistically significant.
Table 4: Comparisons of ABC-SA and ABC variants on 200D problems.

<table>
<thead>
<tr>
<th>Func.</th>
<th>ABC-SA</th>
<th>ABC</th>
<th>GABC</th>
<th>IABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Mean</td>
<td>Std. Dev.</td>
<td>Mean</td>
<td>Std. Dev.</td>
</tr>
<tr>
<td>4.06E+02</td>
<td>3.34E+01</td>
<td>4.32E+02</td>
<td>4.21E+01</td>
<td>=</td>
</tr>
<tr>
<td>F2</td>
<td>1.85E+05</td>
<td>3.30E-05</td>
<td>4.44E-02</td>
<td>2.13E-02</td>
</tr>
<tr>
<td>F3</td>
<td>8.27E-06</td>
<td>2.53E-05</td>
<td>3.17E+01</td>
<td>5.89E+00</td>
</tr>
<tr>
<td>F4</td>
<td>6.05E-10</td>
<td>2.26E-09</td>
<td>5.75E-04</td>
<td>2.58E-03</td>
</tr>
<tr>
<td>F5</td>
<td>6.42E-03</td>
<td>4.93E-04</td>
<td>1.24E-01</td>
<td>1.35E-02</td>
</tr>
<tr>
<td>F6</td>
<td>-8.19E+04</td>
<td>1.00E+02</td>
<td>-7.63E+04</td>
<td>7.61E+02</td>
</tr>
<tr>
<td>F7</td>
<td>8.56E+05</td>
<td>1.81E+04</td>
<td>8.60E+05</td>
<td>1.46E+04</td>
</tr>
<tr>
<td>F8</td>
<td>2.89E+06</td>
<td>9.10E+04</td>
<td>2.91E+06</td>
<td>2.07E+05</td>
</tr>
<tr>
<td>F9</td>
<td>7.63E+11</td>
<td>4.37E+08</td>
<td>8.02E+11</td>
<td>5.05E+09</td>
</tr>
<tr>
<td>F10</td>
<td>3.74E+03</td>
<td>4.19E+01</td>
<td>3.78E+03</td>
<td>9.59E+01</td>
</tr>
<tr>
<td>F11</td>
<td>7.34E+02</td>
<td>4.40E+02</td>
<td>8.82E+03</td>
<td>9.22E+02</td>
</tr>
<tr>
<td>F12</td>
<td>3.79E-11</td>
<td>4.90E-13</td>
<td>1.91E-09</td>
<td>9.41E-14</td>
</tr>
<tr>
<td>F13</td>
<td>9.55E-18</td>
<td>1.04E-18</td>
<td>5.02E-15</td>
<td>1.55E-15</td>
</tr>
</tbody>
</table>

Table 5: The comparisons of ABC-SA and DE variants on 30D problems.

<table>
<thead>
<tr>
<th>Func.</th>
<th>Max.FE</th>
<th>ABC-SA</th>
<th>jDE</th>
<th>JADE</th>
<th>SaDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Mean</td>
<td>Std. Dev.</td>
<td>Mean</td>
<td>Std. Dev.</td>
<td>Mean</td>
</tr>
<tr>
<td>300,000</td>
<td>1.19E-01</td>
<td>7.91E-02</td>
<td>1.30E+01</td>
<td>1.40E+01</td>
<td>3.20E-01</td>
</tr>
<tr>
<td>F2</td>
<td>50,000</td>
<td>3.01E-09</td>
<td>7.44E-10</td>
<td>2.37E-04</td>
<td>7.10E-05</td>
</tr>
<tr>
<td>F3</td>
<td>100,000</td>
<td>2.77E-09</td>
<td>6.09E-10</td>
<td>2.37E-04</td>
<td>7.10E-05</td>
</tr>
<tr>
<td>F4</td>
<td>50,000</td>
<td>7.23E-09</td>
<td>4.11E-09</td>
<td>7.29E-06</td>
<td>1.05E-05</td>
</tr>
<tr>
<td>F6</td>
<td>100,000</td>
<td>8.65E-11</td>
<td>4.94E-11</td>
<td>1.70E-10</td>
<td>2.62E-10</td>
</tr>
<tr>
<td>F11</td>
<td>10,000</td>
<td>4.41E+00</td>
<td>3.90E+00</td>
<td>6.13E+02</td>
<td>1.72E+02</td>
</tr>
<tr>
<td>F12</td>
<td>50,000</td>
<td>1.41E-10</td>
<td>7.23E-11</td>
<td>1.80E-05</td>
<td>1.42E-05</td>
</tr>
<tr>
<td>F13</td>
<td>300,000</td>
<td>4.99E-11</td>
<td>2.49E-10</td>
<td>6.08E-10</td>
<td>8.36E-10</td>
</tr>
</tbody>
</table>

are obtained by ABC and ABC-SA on F1 (50D and 100D), F4 (100D and 200D), F7 (all dimensions), F8 200D, F10 50D, and F11 50D. Moreover, on F1 50D, F6 50D, F4 (100D and 200D), F7 (50D and 100D), F8 200D, F10 (50D and 100D), F11 50D, and F13 100D, GABC and ABC-SA perform statistically similar. Further, ABC-SA and IABC perform better convergence behavior on the majority of test cases are given in Figure 3. According to the figure, ABC-SA shows the convergence curves of some benchmark problems are given in Figure 3. According to the figure, ABC-SA shows better convergence behavior on the majority of test cases when compared to ABC, GABC, and IABC.

Furthermore, mean acceptance rate curves for solution acceptance rule in ABC-SA framework are given in Figure 4 and the acceptance rate is determined as follows:

\[
\text{Acceptance rate} = \frac{\text{number of accepted worse solutions}}{\text{total number of worse solutions}}.
\]

The curves in Figure 4 clearly coincide with the acceptance probability curve given in Figure 2. Figure 4 also shows the nonlinear decreasing of acceptance rate throughout the search process.

4.4 Comparison with PSO and DE Variants. The performance of ABC-SA is also tested against novel and powerful variants of DE and PSO. The competitor algorithms are self-adapting DE (jDE) [41], adaptive DE with optional external archive (JADE) [42], self-adapting DE (SaDE) [43], comprehensive learning PSO (CLPSO) [44], self-organizing hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC) [45], and fully informed particle swarm (FIPS) [46]. The results of these algorithms are taken directly from corresponding studies. The experimental results for 30D problems are shown in Tables 5 and 6 for DE and PSO variants, respectively. Some of the benchmarks problems are not included in the comparisons, since results on these problems were not reported in the corresponding studies. The previous parameter setting for ABC-SA is used, but this time maximum function evaluation number (Max.FE) is employed as the stopping criteria. Since the results of the competitor algorithms are taken directly from corresponding studies, statistical significance tests could not be applied. Therefore, in this part of the analysis, mean and standard deviation are used to compare the performance of the different algorithms.
Figure 3: Convergence curves for ABC-SA and ABC variants.
Table 6: The comparisons of ABC-SA and PSO variants on 3D problems.

<table>
<thead>
<tr>
<th>Func.</th>
<th>Max.FE</th>
<th>ABC-SA</th>
<th>FIPS</th>
<th>HPSO-TVAC</th>
<th>CLPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Std. Dev.</td>
<td>Mean</td>
<td>Std. Dev.</td>
</tr>
<tr>
<td>F1</td>
<td>200,000</td>
<td>7.61E+00</td>
<td>2.18E-01</td>
<td>2.51E+01</td>
<td>5.10E-01</td>
</tr>
<tr>
<td>F2</td>
<td>200,000</td>
<td>4.01E-13</td>
<td>1.05E-13</td>
<td>2.33E-07</td>
<td>7.19E-08</td>
</tr>
<tr>
<td>F3</td>
<td>200,000</td>
<td>6.91E-10</td>
<td>9.44E-11</td>
<td>6.51E+01</td>
<td>1.33E+01</td>
</tr>
<tr>
<td>F4</td>
<td>200,000</td>
<td>8.89E-12</td>
<td>6.56E-13</td>
<td>9.01E-12</td>
<td>1.84E-11</td>
</tr>
<tr>
<td>F6</td>
<td>200,000</td>
<td>1.79E-12</td>
<td>8.89E-11</td>
<td>9.93E+02</td>
<td>5.09E+02</td>
</tr>
<tr>
<td>F11</td>
<td>200,000</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>F12</td>
<td>200,000</td>
<td>5.03E-23</td>
<td>1.44E-24</td>
<td>2.70E-14</td>
<td>1.57E-14</td>
</tr>
</tbody>
</table>

Conflicts of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References


Submit your manuscripts at
http://www.hindawi.com